UCRL-TR-228592

Vibration response for a multilayer cylinder

D. H. Chambers

March 2, 2007

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Vibration response for a multilayer cylinder

David H. Chambers

February 28, 2007

1 Analytical solution

Consider a circular cylinder composed of N concentric layers of materials. Each layer is characterized by its material density ρ_n, and elastic moduli λ_n and μ_n ($n = 1, \dots, N$). The interfaces between layers are specifed by the radii $R_0,\ R_1, \dots,\ R_n$, with R_0 the inner surface and R_N the outer surface of the cylinder. For each layer, the radial displacement field $u_n(r,\theta)$ and tangential displacement field $v_n(r,\theta)$ can be expressed using two elastic potentials $\phi_n(r,\theta)$ and $\psi_n(r,\theta)$:

$$\begin{aligned} u_n(r,\theta) =& \frac{\partial \phi_n}{\partial r} + \frac{1}{r}\frac{\partial \psi_n}{\partial \theta} \,, \\ v_n(r,\theta) =& \frac{1}{r}\frac{\partial \phi_n}{\partial \theta} - \frac{\partial \psi_n}{\partial r} \,. \end{aligned} \tag{1}$$

The radial normal stress $\sigma_{rr}^{(n)}(r,\theta)$ and shear stress $\sigma_{r\theta}^{(n)}(r,\theta)$ are given by

$$\begin{aligned} \sigma_{rr}^{(n)}(r,\theta) =& (\lambda_n + 2\mu_n)\frac{\partial^2 \phi_n}{\partial r^2} + \frac{\lambda_n}{r}\left(\frac{\partial \phi_n}{\partial r} + \frac{1}{r}\frac{\partial^2 \phi_n}{\partial \theta^2}\right) + \frac{2\mu_n}{r}\left(\frac{\partial}{\partial r} - \frac{1}{r}\right)\frac{\partial \psi_n}{\partial \theta} \,, \\ \sigma_{r\theta}^{(n)}(r,\theta) =& \frac{2mu_n}{r}\left(\frac{\partial}{\partial r} - \frac{1}{r}\right)\frac{\partial \phi_n}{\partial \theta} + \mu_n\left(-\frac{\partial^2 \psi_n}{\partial r^2} + \frac{1}{r}\frac{\partial \psi_n}{\partial r} + \frac{1}{r^2}\frac{\partial^2 \psi_n}{\partial \theta^2}\right) \,. \end{aligned} \tag{2}$$

The potentials are solutions to two Helmholtz equations:

$$\nabla^2 \phi_n + k_{Ln}^2 \phi_n = 0 \,, \qquad k_{Ln}^2 = \frac{\omega^2}{c_{Ln}^2} = \frac{\rho_n \omega^2}{\lambda_n + 2\mu_n} \,; \tag{3}$$

$$\nabla^2 \psi_n + k_{Tn}^2 \psi_n = 0 \,, \qquad k_{Tn}^2 = \frac{\omega^2}{c_{Tn}^2} = \frac{\rho_n \omega^2}{\mu_n} \,. \tag{4}$$

For cylindrical geometry, the solutions to the Helmholtz equations can be written as Fourier series in the angular coordinate θ:

$$\phi_n(r,\theta) = \sum_{m=0}^{\infty} \varepsilon_m (-1)^m \left[A_n^{(m)} J_m(k_{Ln} r) + B_n^{(m)} Y_m(k_{Lm} r)\right] \cos(m\theta) \,, \tag{5}$$

$$\psi_n(r,\theta) = \sum_{m=0}^{\infty} \varepsilon_m (-1)^m \left[D_n^{(m)} J_m(k_{Tn} r) + E_n^{(m)} Y_m(k_{Tn} r)\right] \sin(m\theta) \,, \tag{6}$$

where

$$\varepsilon_m = \begin{cases} 1 \,, & m = 0 \\ 2 \,, & m \geq 1 \end{cases} \,. \tag{7}$$

Equations for the coefficients $A_n^{(m)}$, $B_n^{(m)}$, $D_n^{(m)}$, and $E_n^{(m)}$, are obtained by requiring continuity of displacement and stress across each interface ($r = R_1,\ R_2, \dots,\ R_{N-1}$) and by specifying the tractions on the inner

$(r = R_0)$ and outer $(r = R_N)$ surfaces. These conditions can be applied independently for each order m in the Fourier series. The final matrix equation for the coefficients of the mth mode is $\mathbf{Z}^{(m)} \cdot \mathbf{C}^{(m)} = \mathbf{b}^{(m)}$, where $\mathbf{C}^{(m)}$ is a vector containing the coefficients, $\mathbf{b}^{(m)}$ is a vector determined from the applied tractions, and $\mathbf{Z}^{(m)}$ is a matrix consisting of Bessel functions evaluated at the interfaces.

The vector $\mathbf{C}^{(m)}$ can be constructed by stacking N vectors $\mathbf{C}_n^{(m)}$:

$$\mathbf{C}^{(m)} = \begin{bmatrix} \mathbf{C}_1^{(m)} \\ \mathbf{C}_2^{(m)} \\ \vdots \\ \mathbf{C}_N^{(m)} \end{bmatrix} , \qquad \mathbf{C}_n^{(m)} = \begin{bmatrix} A_n^{(m)} \\ B_n^{(m)} \\ D_n^{(m)} \\ E_n^{(m)} \end{bmatrix} . \tag{8}$$

The matrix $\mathbf{Z}^{(m)}$ can be written in terms of the 4×4 matrices $\mathbf{S}_n^{(m)}(r)$ and the 2×4 matrices $\mathbf{F}_n^{(m)}(r)$:

$$\mathbf{Z}^{(m)} = \begin{bmatrix} \mathbf{F}_1^{(m)}(R_0) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{S}_1^{(m)}(R_1) & -\mathbf{S}_2^{(m)}(R_1) & \ddots & \vdots \\ \mathbf{0} & \ddots & \ddots & \mathbf{0} \\ \vdots & \ddots & \mathbf{S}_{N-1}^{(m)}(R_{N-1}) & -\mathbf{S}_N^{(m)}(R_{N-1}) \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{F}_N^{(m)}(R_N) \end{bmatrix} . \tag{9}$$

The $\mathbf{S}_n^{(m)}(r)$ matrices are composed of the $\mathbf{F}_n^{(m)}(r)$ matrices and another 2×4 matrix $\mathbf{G}_n^{(m)}(r)$:

$$\mathbf{S}_n^{(m)}(r) = \begin{bmatrix} \mathbf{F}_n^{(m)}(r) \\ \mathbf{G}_n^{(m)}(r) \end{bmatrix} , \tag{10}$$

where

$$\mathbf{G}_n^{(m)}(r) = \begin{bmatrix} k_{Ln} J'_m(k_{Ln} r) & k_{Ln} Y'_m(k_{Ln} r) & \frac{m}{r} J_m(k_{Tn} r) & \frac{m}{r} Y_m(k_{Tn} r) \\ \frac{m}{r} J_m(k_{Ln} r) & \frac{m}{r} Y_m(k_{Ln} r) & k_{Tn} J'_m(k_{Tn} r) & k_{Tn} Y'_m(k_{Tn} r) \end{bmatrix} . \tag{11}$$

Finally, the $\mathbf{F}_n^{(m)}(r)$ matrix is given by

$$\mathbf{F}_n^{(m)}(r) = \begin{bmatrix} F_{11n}^{(m)}(k_{Ln} r) & F_{12n}^{(m)}(k_{Ln} r) & \frac{2\rho_n \omega c_{Tn} m}{r} g_m^{(1)}(k_{Tn} r) & \frac{2\rho_n \omega c_{Tn} m}{r} g_m^{(2)}(k_{Tn} r) \\ \frac{2\rho_n \omega c_{Tn}^2 m}{c_{Ln} r} g_m^{(1)}(k_{Ln} r) & \frac{2\rho_n \omega c_{Tn}^2 m}{c_{Ln} r} g_m^{(2)}(k_{Ln} r) & F_{23n}^{(m)}(k_{Tn} r) & F_{24n}^{(m)}(k_{Tn} r) \end{bmatrix} , \tag{12}$$

with

$$\begin{bmatrix} F_{11n}^{(m)}(k_{Ln} r) & F_{12n}^{(m)}(k_{Ln} r) \end{bmatrix} = -\rho_n \omega^2 \begin{bmatrix} J_m(k_{Ln} r) & Y_m(k_{Ln} r) \end{bmatrix} - \frac{2\rho_n \omega c_{Tn}^2}{c_{Ln} r} \begin{bmatrix} f_m^{(1)}(k_{Ln} r) & f_m^{(2)}(k_{Ln} r) \end{bmatrix} , \tag{13}$$

and

$$\begin{bmatrix} F_{23n}^{(m)}(k_{Tn} r) & F_{24n}^{(m)}(k_{Tn} r) \end{bmatrix} = \rho_n \omega^2 \begin{bmatrix} J_m(k_{Tn} r) & Y_m(k_{Tn} r) \end{bmatrix} + \frac{2\rho_n \omega c_{Tn}}{r} \begin{bmatrix} f_m^{(1)}(k_{Tn} r) & f_m^{(2)}(k_{Tn} r) \end{bmatrix} . \tag{14}$$

The functions $f_m^{(1)}$, $f_m^{(2)}$, $g_m^{(1)}$, and $g_m^{(2)}$ are combinations of Bessel functions given below:

$$\begin{aligned} f_m^{(1)}(x) &= J'_m(x) - \frac{m^2}{x} J_m(x) \\ f_m^{(2)}(x) &= Y'_m(x) - \frac{m^2}{x} Y_m(x) \\ g_m^{(1)}(x) &= J'_m(x) - \frac{1}{x} J_m(x) \\ g_m^{(2)}(x) &= J'_m(x) - \frac{1}{x} J_m(x) \end{aligned} \tag{15}$$

This completes the specification of the matrix $\mathbf{Z}^{(m)}$.

The last part of the solution is determining the forcing vector $\mathbf{b}^{(m)}$ for each mode. We will consider two cases, normal point load on the inside surface $r = R_0$, and normal point load on the outer surface $r = R_N$. In either case we can assume the load is applied at angular position $\theta = 0$ with no loss of generality. The mathematical expression for the normal point load is

$$\sigma_{rr}(R,\theta) = -\frac{1}{R}\delta(\theta) = -\frac{1}{2\pi R}\sum_{m=0}^{\infty}\varepsilon_m \cos(m\theta) \,, \tag{16}$$

with $R = R_0$ for the inside surface, or $R = R_N$ for the outside surface. After substituting the Fourier series (5) into the expression for the normal stress in (2), then equating it with (16) above, we obtain the following for the forcing vectors $\mathbf{b}_0^{(m)}$ and $\mathbf{b}_N^{(m)}$ for the inside and outside surfaces:

$$\mathbf{b}_0^{(m)} = -\frac{i^m}{2\pi R_0}\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix} \quad , \qquad \mathbf{b}_N^{(m)} = -\frac{i^m}{2\pi R_N}\begin{bmatrix}0\\\vdots\\0\\1\\0\end{bmatrix} \,. \tag{17}$$

Once the coefficients for each mode are obtained by solving $\mathbf{Z}^{(m)}\cdot\mathbf{C}^{(m)} = \mathbf{b}^{(m)}$, we would like to calculate the normal displacement on the inner and outer surfaces. Using the series expansions (5) for the potentials and their relationships to the displacements (1) we obtain the following:

$$\begin{aligned} u_0(\theta) = u(R_0,\theta) = k_{L0}\sum_{m=0}^{\infty}\varepsilon_m(-i)^m \Big\{ & J_m'(k_{L0}R_0)A_0^{(m)} + Y_m'(k_{L0}R_0)B_0^{(m)} \\ & + \frac{m}{k_{L0}R_0}\left[J_m(k_{T0}R_0)D_0^{(m)} + \frac{m}{R_0}Y_m(k_{T0}R_0)E_0^{(m)}\right]\Big\} \,, \end{aligned} \tag{18}$$

$$\begin{aligned} u_N(\theta) = u(R_N,\theta) = k_{LN}\sum_{m=0}^{\infty}\varepsilon_m(-i)^m \Big\{ & J_m'(k_{LN}R_N)A_N^{(m)} + Y_m'(k_{LN}R_N)B_N^{(m)} \\ & + \frac{m}{k_{LN}R_N}\left[J_m(k_{TN}R_N)D_N^{(m)} + \frac{m}{R_N}Y_m(k_{TN}R_N)E_N^{(m)}\right]\Big\} \,. \end{aligned} \tag{19}$$

Suppose we sample the normal displacements at angular positions $\theta = \theta_n = 2\pi n/N$ for $n = 1,\ldots,N-1$, then we can extract the *jth* term ($j \le N$) of either (18) or (19) by evaluating the weighted sums

$$u_{0j} = \sum_{n=0}^{N-1} u_0(\theta_n)\cos\left(\frac{2\pi jn}{N}\right) \tag{20}$$

for the inner surface, and

$$u_{Nj} = \sum_{n=0}^{N-1} u_N(\theta_n)\cos\left(\frac{2\pi jn}{N}\right) \tag{21}$$

for the outer surface. These are the quantities are plotted in the text to determine the vibrational response in each example.

Finally, there are two special cases that can occur: a fluid layer ($\mu_n = 0$ for some n), and a solid core cylinder ($R_0 = 0$). In a fluid the shear stress is zero, which eliminates the potential ψ_n, so that $D_n^{(m)} = E_n^{(m)} = 0$ identically. In addition, the shear stress in adjacent layers must go to zero at the interfaces with the fluid layer ($\sigma_{r\theta}^{(n-1)}(R_{n-1},\theta) = 0$ and $\sigma_{r\theta}^{(n+1)}(R_n,\theta) = 0$). Similarly, there is no continuity condition for the tangential displacements (v_{n-1} and v_{n+1}) in adjacent layers. Thus we can eliminate the columns of

$Z^{(m)}$ corresponding to $D_n^{(m)}$ and $E_n^{(m)}$, along with the rows of $Z^{(m)}$ containing the first and third rows of $\mathbf{S}_n^{(m)}$. This modification can be done for each fluid layer to determine the final form for $Z^{(m)}$.

If the cylinder has a solid core ($R_0 = 0$), then the condition that the potentials must be finite at $r = 0$ forces $B_1^{(m)} = E_1^{(m)} = 0$. It also eliminates the free surface condition for the innermost layer. We can then eliminate the first two rows of $Z^{(m)}$ along with the second and fourth columns which correspond to $B_1^{(m)}$ and $E_1^{(m)}$ to determine the final form for $Z^{(m)}$.

2 Acknowledgements / Disclaimer

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

```
function [ampin,ampout] = measure_cyl(layer_parms,w,Z,maxorder,src_vec)
% function [ampin,ampout] = measure_cyl(layer_parms,w,Z,maxorder,src_vec)
% This function calculates the "measured" modal amplitudes
% for a cylinder represented by the matrix Z, due to a
% source. Inputs are:
%   layer_parms:     structure of layer parameters
%              .R:       interface radii
%              .mu:      shear modulus (Lame)
%              .lambda:  compression modulus (Lame)
%              .rho:     density
%   w:        angular frequency
%   Z:        matrix for modal coefficients
%   maxorder: maximum order of angular dependence (0:maxorder)
%   src_vec:  source vector
% Note ampin = 0 if cylinder is solid (R(1)=0).
% Created 7/11/06

    R = layer_parms.R;
    mu = layer_parms.mu;
    imu0 = find(abs(mu)==0);
    imu0c = find(abs(mu)>0);
    lambda = layer_parms.lambda;
    rho = layer_parms.rho;
    nlay = length(rho);
    nZ0 = 4*nlay;
    nR = nlay+1;
    cL = sqrt((lambda+2*mu)./rho);
    cT = sqrt(mu./rho);
    kL = w./cL;
    kT = zeros(nlay,1);
    if length(imu0)>0
        kT(imu0) = 1.e8;
        if length(imu0c)>0
            kT(imu0c) = w./cT(imu0c);
        end
    else
        kT = w./cT;
    end
    muflag0 = sign(abs(mu));
    m = (0:maxorder)';
    mmax = maxorder+1;
    JL = zeros(mmax,1);
    JT = zeros(mmax,1);
    YL = zeros(mmax,1);
    YT = zeros(mmax,1);
    JpL = zeros(mmax,1);
    YpL = zeros(mmax,1);
    JpT = zeros(mmax,1);
    YpT = zeros(mmax,1);
    gi = zeros(mmax,4);
    go = zeros(mmax,4);
    ampin = zeros(mmax,1);
    ampout = zeros(mmax,1);

    argLo = kL(nlay)*R(nR);
    argTo = kT(nlay)*R(nR);
    [JLo,YLo,JpLo,YpLo] = make_bessel(maxorder,argLo,0);
    [JTo,YTo,JpTo,YpTo] = make_bessel(maxorder,argTo,0);
    go(:,1:2) = kL(nlay)*[JpLo YpLo];
    go(:,3) = muflag0(nlay)*m.*JTo/R(nR);
    go(:,4) = muflag0(nlay)*m.*YTo/R(nR);
    if R(1)>0
        argLi = kL(1)*R(1);
        argTi = kT(1)*R(1);
        [JLi,YLi,JpLi,YpLi] = make_bessel(maxorder,argLi,0);
        [JTi,YTi,JpTi,YpTi] = make_bessel(maxorder,argTi,0);
        gi(:,1:2) = kL(1)*[JpLi YpLi];
        gi(:,3) = muflag0(1)*m.*JTi/R(1);
        gi(:,4) = muflag0(1)*m.*YTi/R(1);
    end

    for jm = 1:mmax
        tmp = zeros(nZ0,1);
        Z1 = squeeze(Z(:,:,jm));
        irZ1 = sign(sum(abs(Z1),2));
        itmp = find(irZ1);
        icZ1 = sign(sum(abs(Z1),1));
        nZ = sum(irZ1);
        if nZ<nZ0
            Z2 = zeros(nZ,nZ);
            indZ = find(irZ1*icZ1>0);
            Z2 = Z1(indZ);
            Z2 = reshape(Z2,nZ,nZ);
            src_vec2 = src_vec(itmp);
%            if rcond(Z2)<1.e-12
                tmp2 = pinv(Z2)*src_vec2;
%            else
%                tmp2 = Z2\src_vec2;
%            end
            tmp(itmp) = tmp2;
        else
%            if rcond(Z1)<1.e12
                tmp = pinv(Z1)*src_vec;
%            else
%                tmp = Z1\src_vec;
%            end
        end
        ampout(jm) = squeeze(go(jm,:))*tmp(((4*nlay-3):(4*nlay)));
        if R(1)>0
            ampin(jm) = squeeze(gi(jm,:))*tmp(1:4);
        end
    end

% end function
```

```
function Z = make_Z(maxorder,layer_parms,w)
% function Z = make_Z(maxorder,layer_parms,w)
% This function creates the Z matrix for a single frequency from the
% layer parameters. Input variables are:
%   maxorder:       maximum order of angular dependence (0:maxorder)
%   layer_parms:    structure of layer parameters
%              .R:       interface radii
%              .mu:      shear modulus (Lame)
%              .lambda:  compression modulus (Lame)
%              .rho:     density
%   w:              angular frequency
% Created 7/10/06

    R = layer_parms.R;
    mu = layer_parms.mu;
    imu0 = find(abs(mu)==0);
    imu0c = find(abs(mu)>0);
    lambda = layer_parms.lambda;
    rho = layer_parms.rho;
    nlay = length(rho);
    nR = nlay+1;
    cL = sqrt((lambda+2*mu)./rho);
    cT = sqrt(mu./rho);
    kL = w./cL;
    kT = zeros(nlay,1);
    if length(imu0)>0
        kT(imu0) = 1.e8;
        if length(imu0c)>0
            kT(imu0c) = w./cT(imu0c);
        end
    else
        kT = w./cT;
    end
    muflag0 = sign(abs(mu));
    m = (0:maxorder)';
    mmax = maxorder+1;
    JL = zeros(mmax,1);
    JT = zeros(mmax,1);
    YL = zeros(mmax,1);
    YT = zeros(mmax,1);
    JpL = zeros(mmax,1);
    YpL = zeros(mmax,1);
    JpT = zeros(mmax,1);
    YpT = zeros(mmax,1);
    F1L = zeros(mmax,1);
    F2L = zeros(mmax,1);
    G1L = zeros(mmax,1);
    G2T = zeros(mmax,1);
    Z = zeros(4*nlay,4*nlay,mmax);

    argL = kL(1)*R(1);
    argT = kT(1)*R(1);
    [JL,YL,JpL,YpL] = make_bessel(maxorder,argL,0);
    [JT,YT,JpT,YpT] = make_bessel(maxorder,argT,0);
    [F1L,F2L,G1L,G2L] = make_FG(maxorder,JL,YL,JpL,YpL,argL,0);
    [F1T,F2T,G1T,G2T] = make_FG(maxorder,JT,YT,JpT,YpT,argT,0);
    a1 = rho(1)*w^2;
```

```
if R(1)==0
    a2 = 0;
    a3 = 0;
else
    a2 = 2*mu(1)*kL(1)/R(1);
    a3 = 2*muflag0(1)*rho(1)*w^2/(kT(1)*R(1));
end
Z(1,1,:) = reshape(-a1*JL-a2*F1L,1,1,mmax);
Z(1,2,:) = reshape(-a1*YL-a2*F2L,1,1,mmax);
Z(1,3,:) = reshape(a3*m.*G1T,1,1,mmax);
Z(1,4,:) = reshape(a3*m.*G2T,1,1,mmax);
Z(2,1,:) = reshape(-a2*m.*G1L,1,1,mmax);
Z(2,2,:) = reshape(-a2*m.*G2L,1,1,mmax);
Z(2,3,:) = muflag0(1)*reshape(a1*JT+a3*F1T,1,1,mmax);
Z(2,4,:) = muflag0(1)*reshape(a1*YT+a3*F2T,1,1,mmax);

for i=2:nlay
    irow = 4*(i-1)-1;
    icol1 = 4*(i-2)+1;
    icol2 = 4*(i-1)+1;
    muprod0 = muflag0(i-1)*muflag0(i);

    argL = kL(i-1)*R(i);
    argT = kT(i-1)*R(i);
    [JL,YL,JpL,YpL] = make_bessel(maxorder,argL,0);
    [JT,YT,JpT,YpT] = make_bessel(maxorder,argT,0);
    [F1L,F2L,G1L,G2L] = make_FG(maxorder,JL,YL,JpL,YpL,argL,0);
    [F1T,F2T,G1T,G2T] = make_FG(maxorder,JT,YT,JpT,YpT,argT,0);
    a1 = rho(i-1)*w^2;
    a2 = 2*mu(i-1)*kL(i-1)/R(i);
    a3 = 2*muflag0(i-1)*rho(i-1)*w^2/(kT(i-1)*R(i));
    Z(irow,icol1,:) = reshape(-a1*JL-a2*F1L,1,1,mmax);
    Z(irow,icol1+1,:) = reshape(-a1*YL-a2*F2L,1,1,mmax);
    Z(irow,icol1+2,:) = reshape(a3*m.*G1T,1,1,mmax);
    Z(irow,icol1+3,:) = reshape(a3*m.*G2T,1,1,mmax);
    Z(irow+1,icol1,:) = reshape(-a2*m.*G1L,1,1,mmax);
    Z(irow+1,icol1+1,:) = reshape(-a2*m.*G2L,1,1,mmax);
    Z(irow+1,icol1+2,:) = muflag0(i-1)*reshape(a1*JT+a3*F1T,1,1,mmax);
    Z(irow+1,icol1+3,:) = muflag0(i-1)*reshape(a1*YT+a3*F2T,1,1,mmax);
    Z(irow+2,icol1,:) = reshape(kL(i-1)*JpL,1,1,mmax);
    Z(irow+2,icol1+1,:) = reshape(kL(i-1)*YpL,1,1,mmax);
    Z(irow+2,icol1+2,:) = reshape(m.*JT/R(i),1,1,mmax);
    Z(irow+2,icol1+3,:) = reshape(m.*YT/R(i),1,1,mmax);
    Z(irow+3,icol1,:) = muprod0*reshape(m.*JL/R(i),1,1,mmax);
    Z(irow+3,icol1+1,:) = muprod0*reshape(m.*YL/R(i),1,1,mmax);
    Z(irow+3,icol1+2,:) = muprod0*reshape(kT(i-1)*JpT,1,1,mmax);
    Z(irow+3,icol1+3,:) = muprod0*reshape(kT(i-1)*YpT,1,1,mmax);

    argL = kL(i)*R(i);
    argT = kT(i)*R(i);
    [JL,YL,JpL,YpL] = make_bessel(maxorder,argL,0);
    [JT,YT,JpT,YpT] = make_bessel(maxorder,argT,0);
    [F1L,F2L,G1L,G2L] = make_FG(maxorder,JL,YL,JpL,YpL,argL,0);
    [F1T,F2T,G1T,G2T] = make_FG(maxorder,JT,YT,JpT,YpT,argT,0);
    a1 = rho(i)*w^2;
    a2 = 2*mu(i)*kL(i)/R(i);
    a3 = 2*muflag0(i)*rho(i)*w^2/(kT(i)*R(i));
```

```
            Z(irow,icol2,:) = -reshape(-a1*JL-a2*F1L,1,1,mmax);
            Z(irow,icol2+1,:) = -reshape(-a1*YL-a2*F2L,1,1,mmax);
            Z(irow,icol2+2,:) = -reshape(a3*m.*G1T,1,1,mmax);
            Z(irow,icol2+3,:) = -reshape(a3*m.*G2T,1,1,mmax);
            Z(irow+1,icol2,:) = -reshape(-a2*m.*G1L,1,1,mmax);
            Z(irow+1,icol2+1,:) = -reshape(-a2*m.*G2L,1,1,mmax);
            Z(irow+1,icol2+2,:) = -muflag0(i)*reshape(a1*JT+a3*F1T,1,1,mmax);
            Z(irow+1,icol2+3,:) = -muflag0(i)*reshape(a1*YT+a3*F2T,1,1,mmax);
            Z(irow+2,icol2,:) = -reshape(kL(i)*JpL,1,1,mmax);
            Z(irow+2,icol2+1,:) = -reshape(kL(i)*YpL,1,1,mmax);
            Z(irow+2,icol2+2,:) = -reshape(m.*JT/R(i),1,1,mmax);
            Z(irow+2,icol2+3,:) = -reshape(m.*YT/R(i),1,1,mmax);
            Z(irow+3,icol2,:) = -muprod0*reshape(m.*JL/R(i),1,1,mmax);
            Z(irow+3,icol2+1,:) = -muprod0*reshape(m.*YL/R(i),1,1,mmax);
            Z(irow+3,icol2+2,:) = -muprod0*reshape(kT(i)*JpT,1,1,mmax);
            Z(irow+3,icol2+3,:) = -muprod0*reshape(kT(i)*YpT,1,1,mmax);
        end

        irow = 4*nlay-1;
        icol = 4*nlay-3;
        argL = kL(nlay)*R(nR);
        argT = kT(nlay)*R(nR);
        [JL,YL,JpL,YpL] = make_bessel(maxorder,argL,0);
        [JT,YT,JpT,YpT] = make_bessel(maxorder,argT,0);
        [F1L,F2L,G1L,G2L] = make_FG(maxorder,JL,YL,JpL,YpL,argL,0);
        [F1T,F2T,G1T,G2T] = make_FG(maxorder,JT,YT,JpT,YpT,argT,0);
        a1 = rho(nlay)*w^2;
        a2 = 2*mu(nlay)*kL(nlay)/R(nR);
        a3 = 2*muflag0(nlay)*rho(nlay)*w^2/(kT(nlay)*R(nR));
        Z(irow,icol,:) = reshape(-a1*JL-a2*F1L,1,1,mmax);
        Z(irow,icol+1,:) = reshape(-a1*YL-a2*F2L,1,1,mmax);
        Z(irow,icol+2,:) = reshape(a3*m.*G1T,1,1,mmax);
        Z(irow,icol+3,:) = reshape(a3*m.*G2T,1,1,mmax);
        Z(irow+1,icol,:) = reshape(-a2*m.*G1L,1,1,mmax);
        Z(irow+1,icol+1,:) = reshape(-a2*m.*G2L,1,1,mmax);
        Z(irow+1,icol+2,:) = muflag0(nlay)*reshape(a1*JT+a3*F1T,1,1,mmax);
        Z(irow+1,icol+3,:) = muflag0(nlay)*reshape(a1*YT+a3*F2T,1,1,mmax);
        if R(1)==0
            Z(1:2,:,:) = 0;
            Z(:,[2 4],:) = 0;
        end

    % end function
```

```
function rout = cyl_outer_response(w,maxorder,layer_parms)
% function rout = cyl_outer_response(w,maxorder,layer_parms)
% This function calculates the frequency response for each mode up to
% maxorder for a layered cylinder with a unit force applied to the outside.
% Output is vector of modal amplitudes (displacement) on outer surface.
% Inputs are:
%   w:        angular frequency vector
%   maxorder: maximum order of angular dependence (0:maxorder)
%   layer_parms:    structure of layer parameters
%              .R:       interface radii
%              .mu:      shear modulus (Lame)
%              .lambda:  compression modulus (Lame)
%              .rho:     density
% Created 2/23/07 from cyl_freq_response

    nw = length(w);
    mmax = maxorder+1;
    rout = zeros(nw,mmax);
    nlayers = length(layer_parms.mu);
    src_vec = zeros(4*nlayers,1);
    src_vec(4*nlayers-1,1) = 1;
    for i=1:nw
        Z = make_Z(maxorder,layer_parms,w(i));
        [ampin,ampout] = measure_cyl(layer_parms,w(i),Z,maxorder,src_vec);
        rout(i,:) = ampout';
    end

% end function
```

```
function [F1,F2,G1,G2] = make_FG(maxorder,J,Y,dJ,dY,x,zeroflag)
% function [F1,F2,G1,G2] = make_FG(maxorder,J,Y,dJ,dY,x,zeroflag)
% This function creates the F1, F2, G1, G2 functions from the
% Bessel functions and derivatives. If zeroflag is present set
% F2 and G2 to zero when x = 0.

      m = (0:maxorder)';
      msq = m.^2;
    mlen = maxorder+1;
    F1 = zeros(mlen,1);
    F2 = zeros(mlen,1);
    G1 = zeros(mlen,1);
    G2 = zeros(mlen,1);
    switch maxorder
        case 0
            if x==0
                if nargin==7
                    Yovx = 0;
                    Jovx = 0;
                else
                    Yovx = Y/x;
                    Jovx = J/x;
                end
            else
                mJovx = 0;
                mYovx = 0;
                Jovx = J/x;
                Yovx = Y/x;
            end
        case 1
            if x==0
                mJovx = [0 ; .5];
                if nargin==7
                    mYovx = [0 ; 0];
                    Jovx = [0 ; 0];
                    Yovx = [0 ; 0];
                else
                    mYovx = [0 ; Y/x;];
                    Jovx = J/x;
                    Yovx = Y/x;
                end
            else
                mJovx = m.^2.*J/x;
                mYovx = m.^2.*Y/x;
                Jovx = J/x;
                Yovx = Y/x;
            end
        otherwise
            if x==0
                mJovx = [0 ; .5 ; zeros(mlen-2,1)];
                if nargin==7
                    mYovx = zeros(mlen,1);
                    Jovx = zeros(mlen,1);
                    Yovx = zeros(mlen,1);
                else
                    mYovx = [0 ; Y(2:mlen)/x];
                    Jovx = J/x;
                    Yovx = Y/x;
                end
            else
                mJovx = m.^2.*J/x;
                mYovx = m.^2.*Y/x;
                Jovx = J/x;
                Yovx = Y/x;
            end
    end

    F1 = dJ-mJovx;
    F2 = dY-mYovx;
    G1 = dJ-Jovx;
    G2 = dY-Yovx;

%      F1 = dJ-m.^2.*J/x;
%    F2 = dY-m.^2.*Y/x;
%      G1 = dJ-J/x;
%    G2 = dY-Y/x;

% end function
```

```
function [Jn,Yn,dJn,dYn] = make_bessel(maxorder,arg,zeroflag)
% function [Jn,Yn,dJn,dYn] = make_bessel(maxorder,arg,zeroflag)
% This function calculates the Bessel function J, Y, and their
% derivatives (dJ, dY) for integer orders up to maxorder with a
% single argument. This is used to calculate acoustic fields for
% cylindrical objects. If zeroflag is present set Y and dY to
% zero when argument is zero.

      nlen = maxorder+1;
      Jn = besselj((0:(maxorder+1))',arg);
    if nargin==3 & arg==0
        Yn = zeros(size(Jn));
    else
      Yn = bessely((0:(maxorder+1))',arg);
    end
      dJn = zeros(nlen,1);
      dYn = zeros(nlen,1);
      dJn(1) = -Jn(2);
      dYn(1) = -Yn(2);
      dJn(2:nlen) = .5*(Jn(1:(nlen-1))-Jn(3:(nlen+1)));
      dYn(2:nlen) = .5*(Yn(1:(nlen-1))-Yn(3:(nlen+1)));
      Jn = Jn(1:nlen);
      Yn = Yn(1:nlen);

% end function
```